Microsoft®

Introducing BizTalk Server 2006 R2

David Chappell, Chappell & Associates

August 2007

Contents

An Overview of BizTalk Server 2006 R2

No application is an island. In fact, tying systems together has become the norm in most organizations today. Yet connecting software means more than just exchanging bytes. As organizations continue to move toward a service-oriented world, the real goal—creating effective business processes that unite disparate systems into a coherent whole—comes within reach.

BizTalk Server 2006 R2 supports this goal. Like its predecessors, this fifth release in the BizTalk Server line allows connecting diverse applications, then creating, executing, and monitoring process logic that uses those applications. The objective is to help organizations create better automated business processes.

The Challenge: Improving Business Processes

The great majority of modern business processes depend at least in part on software. This software has commonly been created at different times using different technologies on different platforms. Given this, automating those business processes requires connecting diverse systems.

Doing this requires solving many different problems, none of them simple. An effective approach is to use a central integration platform that's capable of drawing together all of the systems used in a business process. This technology must be able to do several things, such as the following:

- Connect to diverse software using a range of different approaches. Web services can be the best choice for some connections, simple file sharing might be better for others, while still others might use message queuing or something else. Connecting with line-of-business (LOB) applications also presents its own unique (and important) problems that must be solved.

- Support the execution of automated processes. Something must host the logic that drives an integrated business process, and an integration platform is an obvious choice for this role. While execution of the complete process is actually spread across the various systems involved, an integration platform can implement the centralized logic that controls the process.

- Make connecting with applications in other organizations as easy as possible. This requires supporting industry standards for cross-organization interactions, such as Electronic Data Interchange (EDI), providing services that help connect to trading partners, and more.

- Allow real-time monitoring of business processes. Along with providing a home for hosting the logic that coordinates a process, an integration platform can also provide a central place for monitoring the state of that process. This kind of business activity monitoring allows information workers—the people who are ultimately most concerned with this process—to keep track of exactly what's going on.

The goal of BizTalk Server 2006 R2 is to help organizations improve their business processes by solving these and other problems. The next section takes a big-picture look at how it does this.

Addressing the Challenge: What BizTalk Server 2006 R2 Provides

It's useful to divide the problem of creating better automated business processes into three broad areas:

Connecting applications within a single organization, commonly referred to as *enterprise application integration (EAI)*. As more organizations move toward service-oriented architecture (SOA), the approach to doing this also becomes increasingly service-oriented.

  Connecting applications in different organizations, typically referred to as *business-to-business (B2B) integration*.

  Supporting the holistic approach to working with automated business processes that's defined by *business process management (BPM)*.

Understanding BizTalk Server 2006 R2 requires a grasp of how it addresses each of these three areas.

Application Integration in a Service-Oriented World

Whether it's viewed through the lens of SOA or from the more traditional perspective of EAI, supporting automated business processes requires integrating applications. Figure 1 shows the core BizTalk Server 2006 R2 technologies for doing this: *messaging* and *orchestration*.
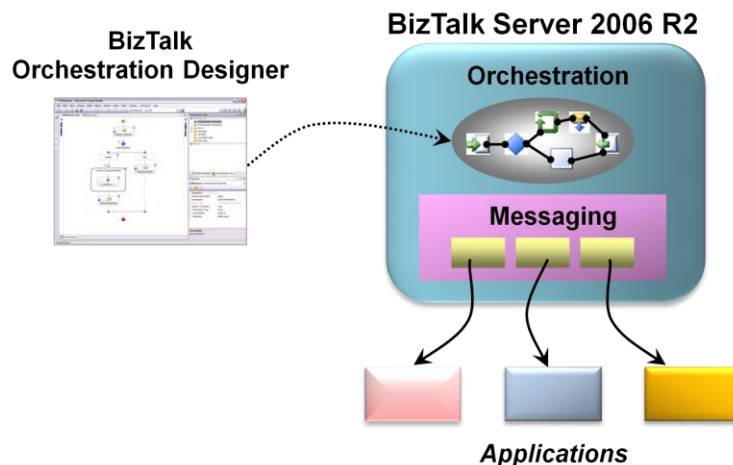

Figure 1: BizTalk Server 2006 R2 provides messaging, orchestration, design tools, and more.

The messaging function contains several parts, one of which is a set of *adapters*. An adapter might implement a particular communication technology, such as Web services, or it might know how to interact with a specific LOB application, such as SAP R/3. With BizTalk Server 2006 R2, Microsoft introduced a new and more general approach to creating these adapters based on Windows Communication Foundation (WCF). Each message is also passed through a *pipeline* that can change it in various ways. And to allow translating among the various formats used by different applications, the messaging function provides *data mapping*. Using various graphical tools, a developer can create pipelines, define maps, and control other aspects of messaging.

While some problems can be solved solely with the messaging function of BizTalk Server 2006 R2, others require creating logic that drives a business process. Orchestrations implement this logic. As Figure 1 shows, developers use a graphical tool called the *BizTalk Orchestration Designer* to create and modify these process definitions.

Developers are key players in the world of BizTalk Server. Yet it's important to understand that business analysts and administrators also have essential roles. A business analyst, for example, might initially define the rules and behaviors that make up a business process. She also determines the flow of the business process, defining what information gets sent to each application and how one business

document is mapped into another. Once the business analyst has defined this process, a developer can create a BizTalk application that implements it. This includes things such as choosing adapters, defining the data mappings for the business documents that will be used, and creating the orchestrations necessary to implement the process logic. An administrator can then deploy the BizTalk application, set up communication among the systems, and perform other tasks. All three roles—business analyst, developer, and administrator—are necessary to create and maintain BizTalk Server 2006 R2 solutions.

Figure 2 shows a simple example of how BizTalk Server 2006 R2 can be applied to an integration problem. In this scenario, an inventory application, perhaps running on an IBM mainframe, notices that the stock of an item is low and so issues a request to order more of that item. This request is sent to a BizTalk Server 2006 R2 orchestration (step 1), which then issues a request to this organization's ERP application requesting a purchase order (step 2). The ERP application, which might be running on a Unix system, sends back the requested PO (step 3), and the orchestration then informs a fulfillment application, perhaps built on Windows using the .NET Framework, that the item should be ordered (step 4).
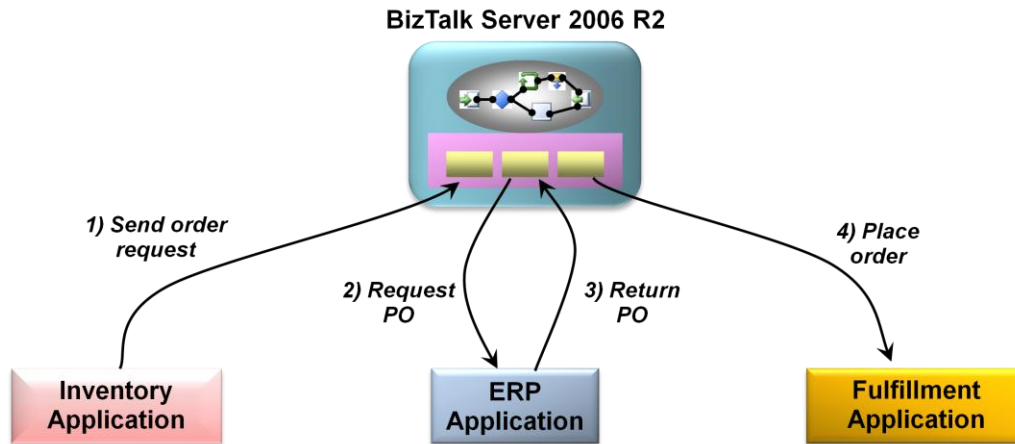

Figure 2: BizTalk Server 2006 R2 can be used to automate a business process that spans multiple applications on different platforms.

In this example, each application might communicate using a different protocol. Accordingly, BizTalk Server 2006 R2 must be able to talk with each one in its native communication style, using the appropriate adapter. Also, notice that no single application is aware of the complete business process. The intelligence required to coordinate all of the software involved is implemented in the BizTalk Server 2006 R2 orchestration.

How does this change in a service-oriented world? One possibility is that the way applications can be accessed becomes more consistent, with Web services a common choice. Another change is that the role of a central integration server might be viewed somewhat differently. A popular term for an integration technology in a service-oriented world is *enterprise service bus (ESB)*, and BizTalk Server 2006 R2 can be used in this style. To help with this approach, Microsoft provides guidance and reference architectures for ESB functionality.

Whether or not an organization takes a service-oriented view, managing integration technology is essential. To allow this, BizTalk Server 2006 R2 includes the *BizTalk Administration console* to let developers and administrators monitor and manage the product. And to help navigate the thicket of logon technologies that diverse applications might use, BizTalk Server 2006 R2 includes an *Enterprise*

Single Sign-on facility. This technology provides a way to map authentication information between Windows and non-Windows systems.

A new addition in BizTalk Server 2006 R2 is support for applications that work with radio frequency identification (RFID). RFID tags can be attached to pallets in a warehouse, products on a shelf, and many other things, then used by applications to track the tagged items. To help create these applications, the latest release of BizTalk Server includes an RFID server.

All of these technologies are useful for connecting applications within a single organization. Most of them can also be applied to connecting applications—and thus automating business processes—across different organizations. The next section looks at how BizTalk Server 2006 R2 supports this goal.

Business-to-Business Integration

Connecting applications within an organization is important, but connecting applications that span organizations often brings at least as much value. Figure 3 shows a simple example of this kind of B2B integration. The customer at the top of the figure runs a BizTalk Server 2006 R2 orchestration that controls a business process. This process allows the customer to purchase items from two supplier organizations. Supplier A also uses BizTalk Server 2006 R2, providing indirect access to its ERP application. Both systems use an appropriate BizTalk adapter to communicate via, say, Web services. Supplier B uses an integration platform from another vendor, connecting to the purchasing organization's BizTalk orchestration using Web services or perhaps some other mechanism.

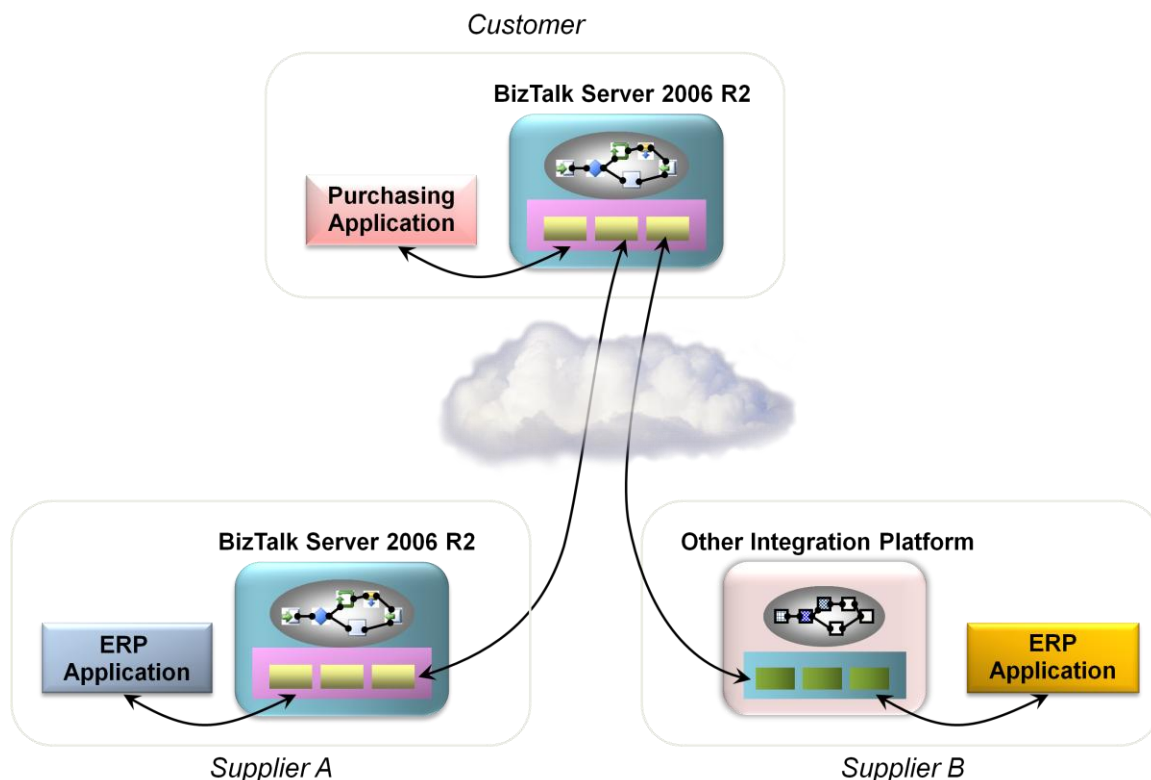**Figure 3: BizTalk Server 2006 R2 can be used to connect applications in different organizations.**

Electronic Data Interchange (EDI) is a fundamental part of B2B communication today. Previous versions of BizTalk Server supported EDI largely through third-party products. With this latest release, Microsoft has chosen to include broad EDI support in the product itself, along with a tool to help

manage relationships with EDI partners. BizTalk Server 2006 R2 also provides *accelerators* to help implement other popular standards, such as RosettaNet, SWIFT, and HL7. Each accelerator includes pre-defined message definitions for the standard, along with relevant guidance and examples.

Business Process Management

Integrating applications into a single automated business process is a fundamental goal of BizTalk Server 2006 R2. It's become common today to view this problem as part of the larger field of business process management. Yet the technology of BPM includes more than integration. As Figure 4 shows, BizTalk Server 2006 R2 also supports two more important BPM technologies: a business rule engine (BRE) and business activity monitoring (BAM).
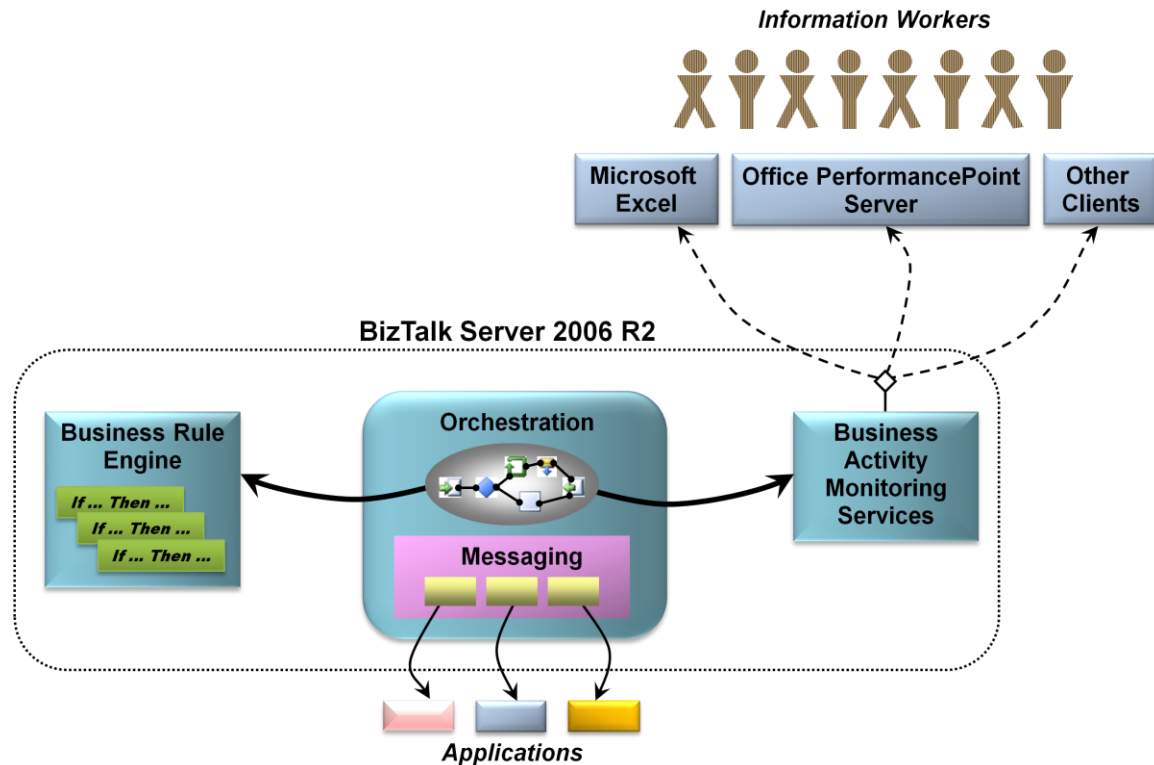

Figure 4: BizTalk Server 2006 R2 includes a BRE and support for BAM.

Like all rules engines, the BRE in BizTalk Server 2006 R2 allows evaluating complex sets of rules. While it's certainly possible to define business logic using the BizTalk Orchestration Designer, some applications require evaluating a complex and often-changed set of rules. Insurance underwriting and loan origination are common examples of this, and there are plenty of others. The goal of the BizTalk BRE is to better support this kind of business process.

However a process is implemented, the people who use it need to know where things stand. How many orders were processed in the last five minutes? How many customers were denied service in the last hour? Providing this kind of real-time data to information workers—not just technicians—can bring substantial business value. The BAM services in BizTalk Server 2006 R2 exist to allow this. As Figure 4 shows, the information BAM provides can be accessed through standard tools, such as Microsoft Excel, Office PerformancePoint Server, and others. This release of BizTalk Server also provides new support for extracting BAM data from applications built using Windows Communication Foundation (WCF) and Windows Workflow Foundation (WF).

Like its predecessors, BizTalk Server 2006 R2 is focused on connecting applications, i.e., on system workflow. A fundamental tenet of BPM, however, is that most business processes include both system and human workflow. To address this, BizTalk Server 2006 R2 can connect to human workflows running on the latest release of Windows SharePoint Services. Accomplished via a SharePoint adapter, this connection lets organizations create automated business processes that include both system workflow and human workflow. In the complex and diverse world of enterprise software today, combining these two approaches is a requirement for many organizations.

BizTalk Server 2006 R2 Fundamentals

Having a broad grasp of the problems it addresses is the first step in understanding BizTalk Server 2006 R2. Going deeper means looking further into the mechanics of how this technology actually works. The place to start is with the basics of message flow, illustrated in Figure 5.
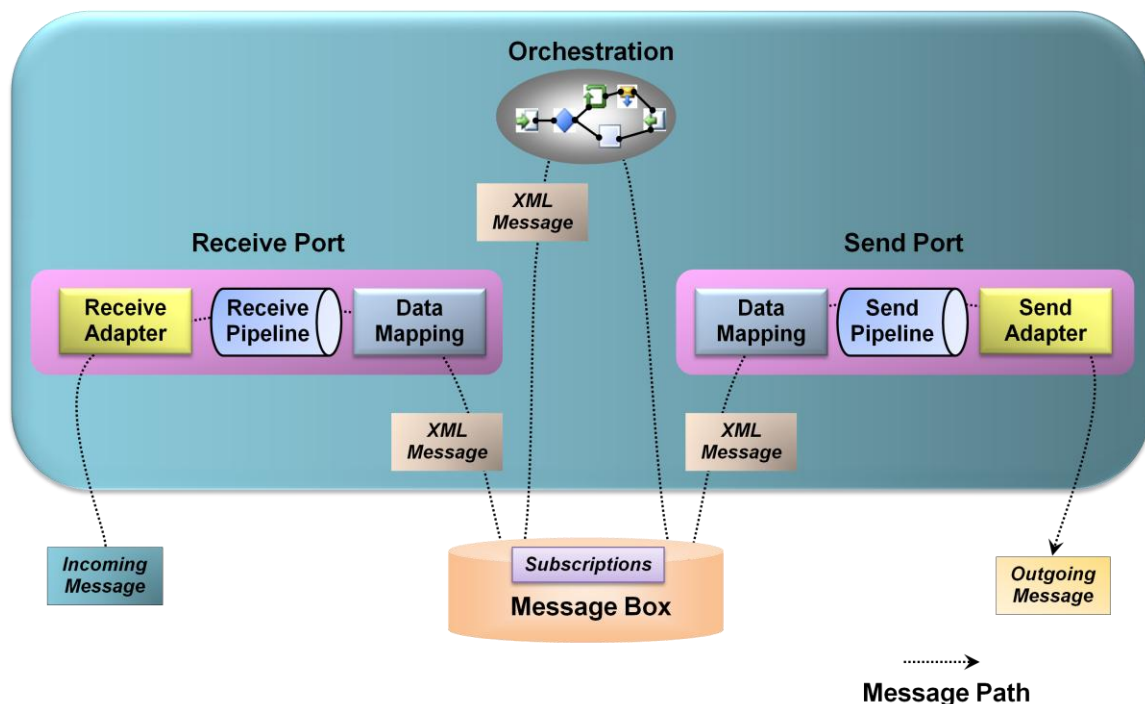
Figure 5: A message is received by a receive port, optionally processed by an orchestration, then sent by a send port.

As the figure shows, a message is received by a *receive port*. Each receive port can have three components:

☐ An *adapter* that knows how to communicate in a specific way;

☐ A *receive pipeline* that does things such as converting the message from its native format into an XML document, validating the message's digital signature, and more;

☐ A *data mapping*, which transforms the message in some useful way.

The message is then delivered into a SQL Server database called the *MessageBox*. From here, it can be read by an orchestration. Orchestrations aren't created by writing code in a language such as C#,

however. Instead, a business analyst or (more likely) a developer uses a graphical tool to create a group of *shapes* that express conditions, loops, and other behavior. And although it's not shown in Figure 5, orchestrations can optionally use the BRE to express complex sets of rules.

Once an orchestration has processed a message, it typically produces another message destined for some other application. This message is placed in the MessageBox, then picked up by a *send port*. A send port can have the same three components as a receive port, and they perform the same functions: mapping the message into its outgoing format, preparing that message for transmission in a *send pipeline*, then actually transmitting it to its destination using an appropriate *send adapter*.

All of this is held together by *subscriptions* stored in the MessageBox. When a message is processed by a receive port, a *message context* is created that contains various *properties* of the message. An orchestration or a send port can subscribe to messages based on the values of these properties. For example, an orchestration might create a subscription that matches all messages of the type "Invoice", or all messages of the type "Invoice" received from the QwickBank corporation, or all messages of the type "Invoice" received from the QwickBank corporation that are for more than $10,000. However it's specified, a subscription returns to its subscriber only those messages that match the criteria that subscription defines. A received message might initiate a new orchestration or it might activate another step in an already running orchestration. When an orchestration sends a message, that message is matched to a send port based on a subscription that port has established.

As this description suggests, a complete solution built on BizTalk Server 2006 R2 contains various parts (sometimes referred to as *artifacts*): orchestrations, pipelines, message schemas, and more. To allow working with these as a single unit, a developer can group them into a *BizTalk application*. Each BizTalk application wraps all of the pieces required for a solution into a single logical unit, making it the fundamental abstraction for management and deployment.

Connecting Systems

BizTalk applications rely on send and receive ports to communicate with other applications. This section takes a closer look at the three components that a port can contain: adapters, pipelines, and data mappings.

Sending and Receiving Messages: Adapters

Interoperating with all kinds of applications on all kinds of systems is a fundamental requirement for integration. BizTalk Server 2006 R2 accomplishes this via adapters. Depending on what a BizTalk application must communicate with, its creator determines which adapters that application should use. He might choose one of the built-in adapters BizTalk Server 2006 R2 provides, use an adapter provided by a third party, or even create a custom adapter.

BizTalk Server 2006 R2 introduces a new approach to creating adapters, along with a new set of adapters built using this approach. Rather than relying on the adapter framework provided in previous releases, these new adapters are built as WCF *channels*. The WCF-based adapters shipped with BizTalk Server 2006 R2 provide support for SOAP, SOAP with WS-* technologies such as WS-Security, and more. Developers can create their own WCF-based adapters using either existing WCF channels or custom channels created for a specific purpose.

Microsoft also provides a BizTalk Adapter Pack that includes WCF-based adapters for SAP, Siebel, and the Oracle database. All of these are created using the *WCF Line-of-Business (LOB) Adapter SDK*, a generalized framework for creating adapters to LOB applications. In fact, adapters created using the WCF LOB Adapter SDK can be used by any .NET Framework application—BizTalk Server isn't required.

BizTalk Server 2006 R2 also includes adapters that implement other commonly used communication mechanisms. The MSMQ Adapter allows sending and receiving messages using Microsoft Message Queuing (MSMQ), for example, while the WebSphere MQ Adapter allows sending and receiving messages using IBM's WebSphere MQ. Similarly, the SMTP Adapter and the POP3 Adapter allow sending and receiving email using these standard protocols.

Other adapters allow interaction via common storage mechanisms. The File Adapter, for instance, allows reading from and writing to files in the Windows file system. Because the applications involved in a business process can often access the same file system, either locally or across a network, exchanging messages through files can be a convenient option. The Windows SharePoint Services Adapter allows accessing and publishing documents stored in SharePoint document libraries, while the SQL Adapter can access information in a SQL Server database. The product also includes adapters for exchanging information using IBM's DB/2 database and Oracle's database.

Another important category of adapters are those that allow connecting to commonly-used business applications. BizTalk Server 2006 R2 provides adapters for a number of these, including SAP R/3, Siebel eBusiness Application, PeopleSoft, JD Edwards OneWorld, and others. The product also provides the *BizTalk Adapters for Host Systems* for connecting to applications running on IBM zSeries mainframes and iSeries mid-range systems.

Whatever receive adapter is used for incoming data, the messages it gets must commonly be processed before they can be accessed by an orchestration. Similarly, outgoing messages produced by an orchestration often need to be processed before they are transmitted by a send adapter. Both kinds of processing depend on pipelines, as described next.

Processing Messages: Pipelines

The applications that support a business process communicate by exchanging various kinds of documents, such as purchase orders, invoices, and many others. For a BizTalk application to implement this process, it must be able to deal correctly with the messages that contain these documents. The processing required to do this can involve multiple steps, and so it's performed by a message pipeline. Incoming messages are processed through a *receive* pipeline, while outgoing messages go through a *send* pipeline.

For example, even though more and more applications understand XML documents, many cannot. Since BizTalk Server 2006 R2 typically works with XML documents internally, it must provide a way to convert other formats to and from XML. Other services may also be required, such as authenticating the sender of a message. To handle these and other tasks in a modular way, a pipeline is constructed from some number of *stages*, each of which contains one or more *components*. Each component handles a particular part of message processing, and BizTalk Server 2006 R2 provides standard components for the most common cases. If these aren't sufficient, developers can also create custom components for both receive and send pipelines.

BizTalk Server 2006 R2 defines a few default pipelines, including a simple receive/send pair that can be used for handling messages that are already expressed in XML. A developer can also create custom pipelines using the *Pipeline Designer*. This tool, which runs inside Visual Studio, provides a graphical interface that allows dragging and dropping components to create pipelines with whatever behavior is required.

Translating Messages: Data Mapping

Pipelines are responsible for converting external documents into and out of an XML representation, if required. Yet it's up to the developer to define what that XML representation looks like, that is, to specify the schema that should be used. Schemas are defined using the XML Schema Definition language (XSD), a powerful but complex way to describe an XML document's structure and the types it can

contain. To make defining XSD schemas easier, BizTalk Server 2006 R2 provides a tool called the *BizTalk Editor*. Rather than creating a schema directly in XSD, this editor lets a developer build a schema by defining its elements in a graphical hierarchy. Existing schemas can also be imported from files or Web services.

Once messages are in a known XML schema, it's possible to map between them. For example, it's common for some of the information in a received document to be transferred to a sent document, perhaps transformed in some way. To allow this, BizTalk Server 2006 R2 lets developers create *maps*. Each map is expressed as a correlation between two XML schemas that defines a relationship between elements in those schemas. The W3C has defined the Extensible Stylesheet Language Transformation (XSLT) as a standard way to express these kinds of transformations between XML schemas, and so maps in BizTalk Server 2006 R2 are implemented as XSLT transformations.

Maps can be used in various ways. Suppose, for instance, that an incoming purchase order needs to have some of its information mapped to an outgoing invoice. A developer might create a map that does this, then invoke that map from a send pipeline—no orchestration is required. In a more complex case that requires more business logic, a map might be invoked from within an orchestration. For example, an order fulfillment process might receive an order for some number of items, then send back a message indicating that the order was declined for some reason. It's possible that information from the order, such as a request identifier and the quantity ordered, should be copied from fields in the received order message into fields in the rejection message.

Maps are just XSLT, so an ambitious developer is free to build them by hand. To make this task easier, BizTalk Server 2006 R2 provides a graphical tool called the BizTalk Mapper. Figure 8 shows how a map for transferring information from a contacts database into a CRM application might look.
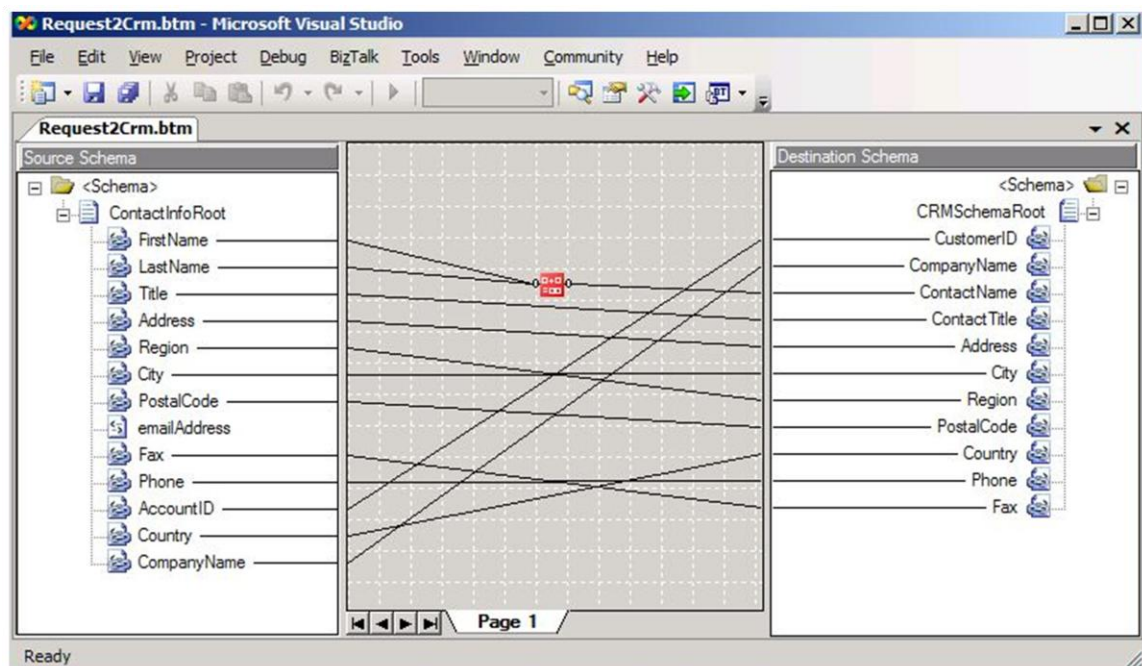

Figure 6: The BizTalk Mapper allows specifying how information in one message should be mapped to another message.

The transformation defined in a map can be simple, such as copying values unchanged from one document to another. Direct data copies like this are expressed using a *link*, which is shown in the BizTalk Mapper as a line connecting the appropriate elements in the source schema with their

counterparts in the destination schema. Most lines in Figure 8 show this kind of connection. More complex transformations are also possible using *functoids*. A functoid is a chunk of executable code that can define arbitrarily complex mappings between XML schemas. As the topmost line in Figure 8 shows, the BizTalk Mapper represents a functoid as a box on the line connecting the elements being transformed. Since some of those transformations are quite common, BizTalk Server 2006 R2 includes a number of standard functoids for performing conversions, mathematical operations, and other tasks.

Having a way to define a document's XML schema is essential, as is a mechanism for mapping information across documents with different schemas. The BizTalk Editor and BizTalk Mapper address these two problems. Yet for many applications, defining schemas and maps isn't enough; business logic must also be specified. How this is done for BizTalk applications is described next.

Defining Business Processes

Sending messages between different systems is a necessary part of solving the problems that BizTalk Server 2006 R2 addresses. Yet while plenty of useful applications can be built using only the product's messaging capabilities, many others also require a way to define and execute process logic. This section describes the technologies BizTalk Server 2006 R2 provides to do this.

Using Orchestrations

In general, it's always possible to implement an automated process directly in a language such as C# or Visual Basic. Yet writing, maintaining, and managing complex business processes created using conventional programming languages can be challenging. Like its predecessors, BizTalk Server 2006 R2 doesn't take this approach. Instead, it allows creating a process's logic graphically. Doing this can be faster than building the process directly in a programming language, and it can also make the process easier to understand, explain, and change.

Successfully creating an automated business process usually requires collaboration between software developers and business people. To help with this, BizTalk Server 2006 R2 provides a tool for each. The developer tool runs inside Visual Studio, an environment in which software professionals feel at home. Most business people don't find Visual Studio especially inviting, however, so BizTalk Server 2006 R2 also provides a subset of the developer tool functionality via an add-in for Visio. An orchestration created in the Visual Studio-based tool can be imported into the Visio-based tool and vice-versa, which helps these two kinds of people work together to automate a business process.

Stripped to its essentials, every business process is a set of actions that together meet some useful business need. The *Orchestration Designer* in BizTalk Server 2006 R2 lets a developer define these actions by connecting together a series of *shapes* in a logical way. Some examples of the shapes available to an orchestration's creator are the following:

The Receive shape, which allows the orchestration to receive messages.

The Send shape, which allows the orchestration to send messages.

The Port shape, which defines how messages are transmitted. Each instance of a port shape is connected to either a Send or Receive shape. Each port also has a type, which defines things such as what kinds of messages this port can receive, and a binding, which determines how a message is sent or received by, for example, specifying a particular URL.

The Decide shape, which represents an if-then-else statement that allows an orchestration to perform different tasks based on Boolean conditions. An *Expression Editor*, part of the Orchestration Designer, can be used to specify this conditional statement.

The Loop shape, which allows performing an action repeatedly while some condition is true.

The Transform shape, which allows transferring information from one document to another, transforming it on the way by invoking maps defined with the BizTalk Mapper.

The Parallel Actions shape, which allows specifying that multiple operations should be performed in parallel rather than in sequence. The shape that follows this one won't be executed until all of the parallel actions have completed.

The Scope shape, which allows grouping operations into transactions and defining exception handlers for error handling. Both traditional atomic transactions and long-running transactions are supported. Unlike atomic transactions, long-running transactions rely on compensating logic rather than rollback to handle unexpected events.

The Message Assignment shape, which allows assigning values to orchestration variables. These variables can be used to store state information used by the orchestration, such as a message being created or a character string.

Figure 9 shows an orchestration created in the Orchestration Designer using a few of these shapes. In this simple example, a message is received, a decision is made based on the content of that message, and one of two paths is executed as a result of that decision. Orchestrations that solve real problems can be significantly more complex than this, of course, and so to help in working with these more complex diagrams, the Orchestration Designer provides the ability to zoom in and out. This lets a developer view only those parts of an orchestration that she's currently interested in. Once a developer has defined an orchestration, the group of shapes and relationships between them is converted into a standard .NET assembly. And it's still possible to add explicit code to an orchestration when necessary by calling a .NET object from inside a shape.
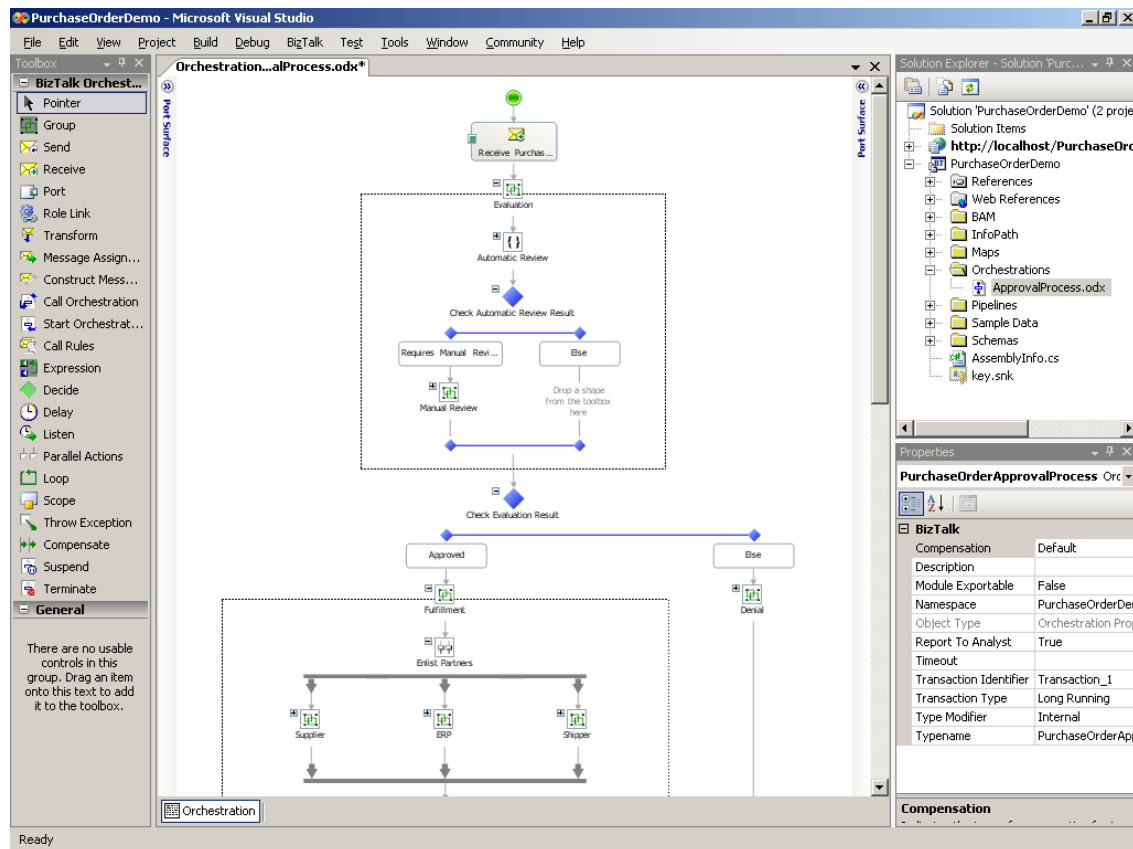
Figure 7: The Orchestration Designer lets a developer create business logic by dragging and dropping shapes from a toolbox onto a design surface.

SOAP-based Web services have had a big impact on application development. To access an external Web service, an orchestration's creator might use the Add Web Reference option in Visual Studio along with the SOAP adapter. BizTalk Server 2006 R2 also includes a *WCF Service Consuming Wizard* that helps developers create orchestrations that consume services exposed via SOAP or any other mechanism supported by WCF. The product provides a *WCF Service Publishing* wizard as well that walks a developer through the steps required to expose one or more of an orchestration's operations as WCF services.

Orchestrations are the fundamental mechanism for creating business processes in BizTalk Server 2006 R2. Yet a significant subset of processes can benefit from an easier way to define and change the business rules they contain. Allowing this is the goal of the Business Rule Engine, as described next.

Using the Business Rule Engine

The Orchestration Designer is a useful tool for defining a business process. Yet some aspects of an orchestration tend to change more often than others. In particular, the decisions embedded in a business process—the business rules—are commonly its most volatile aspect. A manager's spending limit was $100,000 last week, but her promotion bumps this up to $500,000, or a slow-paying customer's maximum allowed order decreases from 100 units to only 10. Why not provide an explicit way to specify and update these rules? To allow this, BizTalk Server 2006 R2 includes the BRE. Developers will most often use the BRE, but it's also possible for more business-oriented users to create and modify sets of business rules using a tool called the *Business Rule Composer*.

The BRE is most useful when a complex set of business rules must be evaluated. Deciding whether to grant a loan, for example, might entail working through a large set of rules based on the customer's credit history, income, and more. Similarly, determining whether to sell life insurance to an applicant depends on a number of things, including the applicant's age, gender, and a myriad of health factors. Expressing all of these rules as conditional statements using, say, an orchestration's Decide shape might be possible, but it's not simple. For rule-intensive processes like these, the BRE can make a developer's life significantly simpler.

The BRE can also make changing rules faster and easier. To see why, think about what's required to change a business rule that's implemented within an orchestration. A developer must first open the orchestration in Visual Studio, modify the appropriate shapes (and perhaps any .NET objects they invoke), then build and deploy the modified assembly. Doing this also requires stopping and re-starting the BizTalk application that includes this orchestration. If instead this business rule is implemented using the BRE, it can be modified without recompiling or restarting anything. All that's required is using the Business Rule Composer to change the desired rule, then redeploying the new set of rules. The change will take effect immediately. And while orchestrations are typically created and maintained by developers, business rules are readable enough that in some cases they can be modified by business analysts without the need to involve more technical people.

The creator of a set of business rules will typically begin by using the Business Rule Composer to define a *vocabulary* for use in specifying those rules. Each term in the vocabulary provides a user-friendly name for some information. For example, a vocabulary might define terms such as Number Shipped or Maximum Quantity of Items or Approval Limit. Each of these terms can be set to a constant or be mapped to a particular element or attribute in some XML schema (and thus in an incoming message) or to the result of a SQL query against some database or even to a value in a .NET object.

Once a vocabulary has been defined, the Business Rule Composer can be used to create *business policies* that use this vocabulary. Each policy can contain one or more business rules. A rule uses the terms defined in some vocabulary together with logical operators such as Greater Than, Less Than, Is Equal To, and others to define how a business process operates. A business rule can define how values contained in a received XML document should affect the values created in an XML document that's sent, or how those received values should affect what value is written in a database, or other things.

Imagine, for instance, a simple vocabulary that defines the term Maximum Allowed Order Quantity, whose value is set to 100, and the term Quantity Requested, whose value is derived from a specified element in received XML documents that correspond to the schema used for placing orders. A business analyst might create a rule stating that if the Quantity Requested in an incoming order is greater than the Maximum Allowed Order Quantity, the order should be rejected, perhaps resulting in an appropriate XML document being sent back to the originator of this order.

To execute a business policy, an orchestration uses a CallRules shape. This shape creates an instance of the BRE, specifies which policy to execute, then passes in the information this policy needs, such as a received XML document. The BRE can also be invoked programmatically via a .NET-based object model, which allows it to be called from applications that don't use BizTalk Server 2006 R2 (although a BizTalk Server license is always required to use the BRE).

Both vocabularies and business rules can be much more complicated—and much more powerful—than the simple examples described here. But the core idea of defining a vocabulary, then defining sets of rules that use that vocabulary is the heart of the Business Rule Engine. The goal is to provide a straightforward way for BizTalk Server 2006 R2 users to create and work with the rules that pervade business processes.

Creating Scalable Configurations

It's possible to install every component of BizTalk Server 2006 R2 on a single machine. Yet it's not hard to imagine situations where this isn't the right solution. Perhaps the number of messages the system must handle is too great for one machine, or maybe redundancy is required to make the system more reliable. To meet requirements like these, the product can be deployed in a number of ways.

A fundamental concept for deploying BizTalk Server is the idea of a *host*. A host can contain various things, including orchestrations, adapters, and pipelines. Hosts are just logical constructs, however. To use them, a BizTalk administrator must cause actual *host instances* to be created. Each host instance is a Windows process, and as Figure 10 shows, it can contain various things. In the example shown here, Machine A is home to two host instances. One contains a receive port, while the other contains the orchestrations P and Q. Machine B runs just one host instance, also containing the two orchestrations P and Q. Machine C, like machine A, is home to two host instances, but neither of them contains an orchestration. Instead, each of these instances contains a different send port. Finally, machine D houses the MessageBox database that's used by all of the host instances in this configuration.
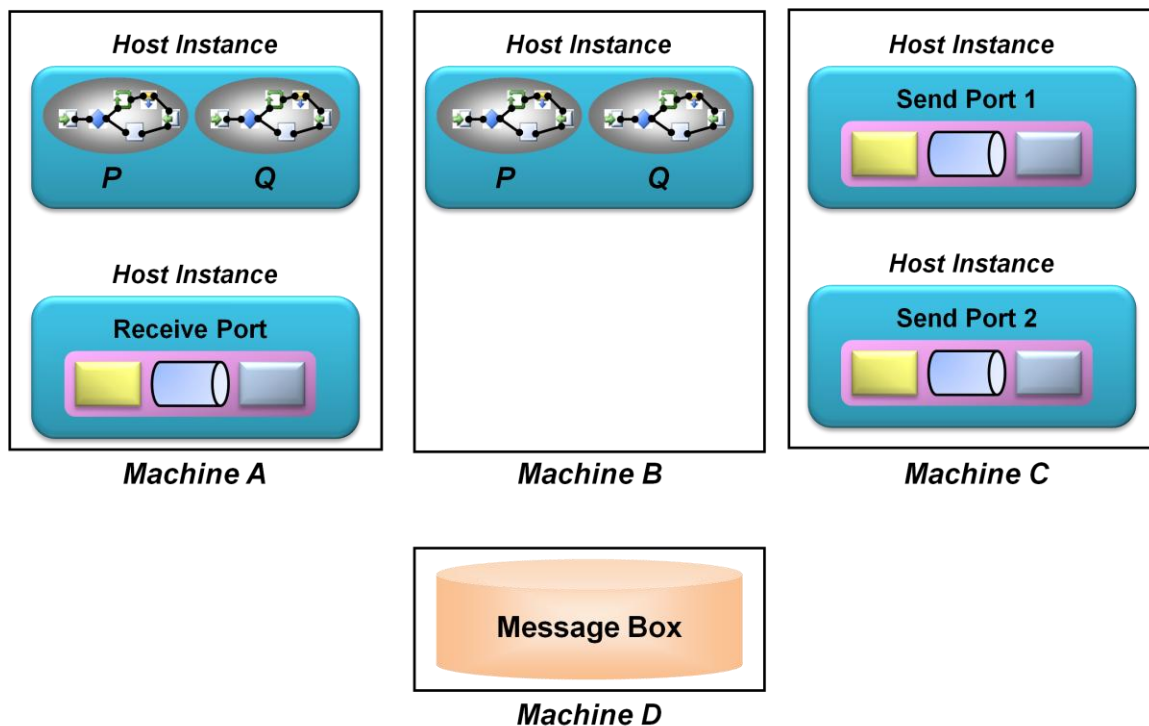

Figure 8: A single BizTalk Server installation can be spread across multiple hosts on multiple machines.

This example illustrates several ways in which hosts might be used. For instance, since both machines A and B are home to the orchestrations P and Q, BizTalk Server 2006 R2 can automatically load balance requests to these orchestrations based on the availability and current load on each machine. This allows a BizTalk application to scale up as needed for high-volume processes. Notice also that machine C contains two different ways to handle outgoing messages, with each perhaps using a different send adapter. And because each host instance is isolated from every other host instance—they're separate Windows processes—it's safer to run code that's not completely trusted, such as a new custom adapter, in a separate instance. It's also worth pointing out that even though this example

contains only a single instance of the MessageBox database, it's possible to replicate this database for greater scalability or cluster it to avoid creating a single point of failure.

Managing BizTalk Applications

Whatever it does and however it's configured, every BizTalk application must be managed. What's the state of running applications right now? How have applications behaved in the past, and how can this knowledge be used to improve the future? This section looks at the tools provided to answer these questions.

Creating and Monitoring Applications

The primary tool for creating, monitoring, and managing BizTalk applications is the *BizTalk Administration console*, a Microsoft Management Console (MMC) snap-in that provides a standard user interface for BizTalk administrators. While this tool gives administrators a number of capabilities, the most important are the ability to do three things:

- Deploy BizTalk applications. Using the BizTalk Administration console, an admin can create a BizTalk application, deploy it to one or more servers, and more.

- Configure BizTalk applications. When a developer creates an orchestration, she works largely in logical terms. To define how BizTalk Server 2006 R2 will communicate with a particular application, for example, the developer can select the WCF SOAP adapter without worrying about the specific URL that will be used. Similarly, she can specify that the send pipeline should include a component that adds a digital signature to outgoing messages without worrying about exactly what key will be used to create this signature. Yet to make the application work, these details must be specified. The BizTalk Administration console allows an admin to create and modify details like these.

- Monitor and manage BizTalk applications. Using the BizTalk Administration console's Group Hub page, an administrator can monitor the operation of BizTalk applications. As Figure 11 shows, information about the status of these applications can be examined in various ways. An administrator can look at currently running applications, for example, suspending and restarting them as necessary. It's also possible to look more closely at individual applications, examining specific messages or other details. And rather than requiring an administrator to search for problems, the Group Hub page uses color-coded indicators to display those problems. This lets administrators take a more proactive approach to application monitoring.
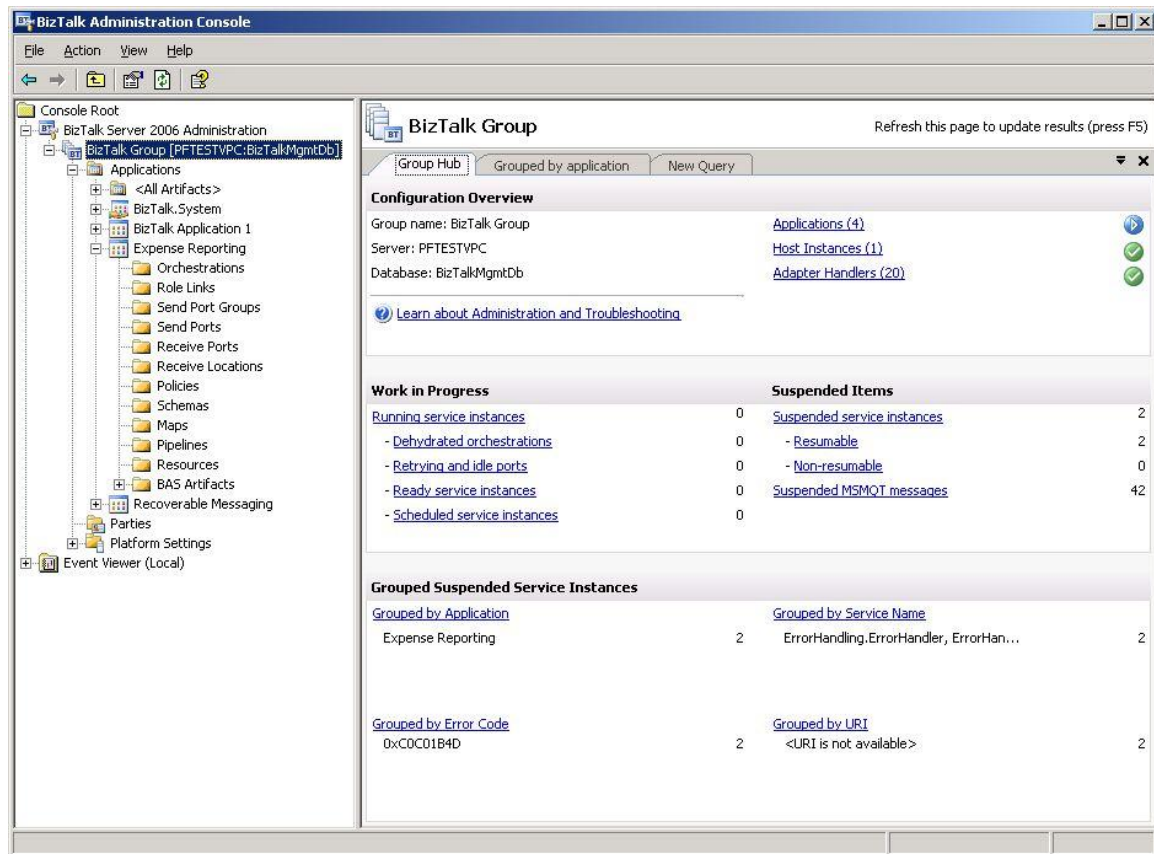
Figure 9: The BizTalk Administration console's Group Hub page lets an administrator monitor and manage running BizTalk applications.

The BizTalk Administration console, which uses BizTalk Server 2006 R2's Configuration database, also provides other services. An administrator can dynamically add machines and specify what hosts should be assigned to them while a BizTalk application is running, for example, without shutting down the application. The Administration console's functions can also be accessed programmatically through Windows Management Instrumentation (WMI), which allows administrators to create scripts that automate management functions. And as with other Microsoft server products, a BizTalk Server 2006 R2 management pack is available, allowing the product itself to be monitored and managed with Microsoft Operations Manager 2005 or System Center Operations Manager 2007

Examining Application History

While the BizTalk Administration console can be used to keep track of current applications, it's also useful to be able to examine historical information about groups of applications. Doing this is the primary purpose of the *Health and Activity Tracking (HAT)* component of BizTalk Server 2006 R2. The HAT tool provides access to aggregated historical information about BizTalk applications running on a system. This information can include when an orchestration starts and ends, when each shape within it is executed, when each of its messages is sent and received, what's in those messages, and more. The HAT tool can be used to examine archived data, looking for patterns and trends in the execution of a process. This information is useful for debugging, answering business questions (such as verifying that a message really was sent to a customer), and keeping ongoing statistics that can be used to improve performance.

Additional BizTalk Server 2006 R2 Technologies

The fundamentals of a BizTalk application revolve around messaging and orchestration. The product provides more than this, however, including business activity monitoring, support for working with RFIDs, and more. This section takes a brief look at each of these technologies.

Business Activity Monitoring

BizTalk Server 2006 R2 provides support for automated business processes that span multiple applications. But once those automated processes have been created, the information workers that use them—business people, not developers—might need to monitor various business-related aspects of the process. To allow this, BizTalk Server 2006 R2 provides BAM.

It's not hard to think of different ways that an information worker might want to look at a business process. A purchasing manager might need to see how many POs are approved and denied each day, for instance, while a sales manager might want an hourly update on what products are being ordered. Meeting these diverse needs requires a general framework for tracking what's going on with a particular business process. This is exactly what BAM provides.

As Figure 12 suggests, it's useful to think of BAM technology in two distinct parts:

☐ An infrastructure for collecting information about in-progress business processes. Because these processes might rely on BizTalk orchestrations and other applications, this infrastructure must be usable with more than just BizTalk Server 2006 R2.

☐ Tools that let information workers access that information. Different people will want to see BAM data in different ways, and so the tools they use might be quite diverse. Some typical examples include dashboards that provide real-time display of critical data, reporting services that present historical trends, and common desktop tools such as spreadsheet applications.
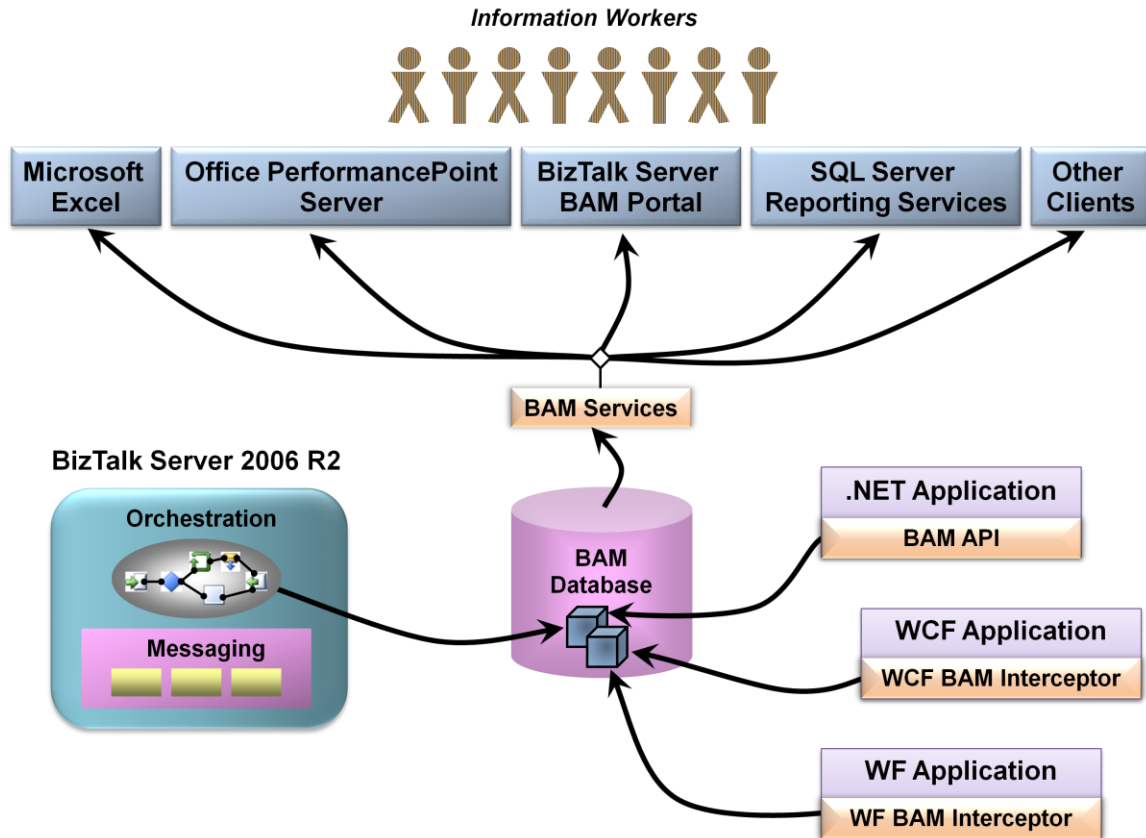
Figure 10: BAM data can be generated by orchestrations and other .NET applications, then used in a variety of ways.

The first of these two aspects, an infrastructure for collecting information about running processes, is provided by BizTalk Server 2006 R2. As Figure 12 shows, BizTalk orchestrations can directly generate BAM data, all of which is sent into a BAM database. Using a tool called the *Tracking Profile Editor*, a developer can configure an orchestration to send the desired information to this database. Via a BizTalk-provided BAM API, this infrastructure can also be used with any application built on the .NET Framework. Along with this general API, BizTalk Server 2006 R2 provides BAM interceptors designed specifically for applications created using WCF and Windows Workflow Foundation (WF).

However it gets to the BAM database, this data is stored in tables and cubes. This information is then accessible via a set of BAM Web services, as shown in Figure 12, and different clients are free to do different things with this information. An Excel user, for instance, might read it into a pivot table, then create a graphical view of the aspects of this process that she wishes to see. (BizTalk Server 2006 R2 provides an Excel add-in to make this easier to do.) This view can be updated as often as necessary, allowing real-time monitoring of the business process.

Other tools can display the data in other ways. Office PerformancePoint Server 2007, for example, might display BAM data generated by one or more business processes as part of a dashboard. The screen shot below shows an illustration of how this might look using PerformancePoint's Business Scorecard Manager.
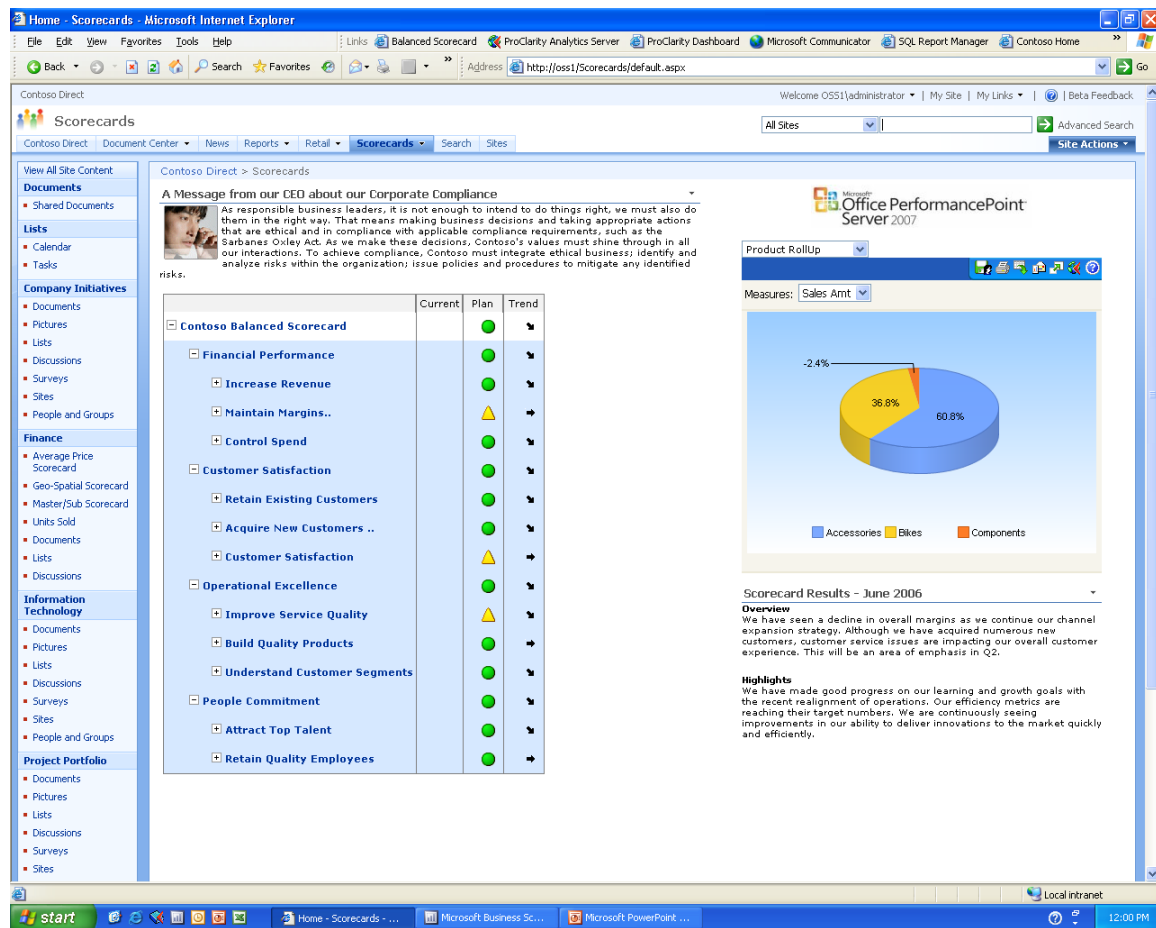
Figure 11: Information displayed via Office PerformancePoint Server 2007 might come from the BAM database

Alternatively, an information worker might use BizTalk Server's BAM portal to select a particular instance of some business process, then choose a specific *BAM view* into the process. Each of these views can give a different perspective, such as graphical depictions of per-product sales trends or current inventory levels or other key performance indicators. The information in these views might be updated every day, every hour, or more frequently. Using the BAM portal, an information worker can define aggregations of data, such as the number of orders filled, canceled, or in progress over the last hour. Implemented as a set of ASP.NET pages, the BAM portal can also be hosted as a Web part inside Windows SharePoint Services.

Using EDI

Business-to-business integration is an important part of what BizTalk Server 2006 R2 does. A large share of B2B connections, somewhere around 80% of the total, relies on the Electronic Data Interchange standards. EDI isn't exactly a state-of-the-art technology—it relies on exchanging character data, not XML documents—but it's nonetheless a bedrock part of B2B integration today. Because of this, BizTalk Server 2006 R2 includes broad EDI support.

Every release of BizTalk Server has offered some support for EDI, and partners have provided more. With BizTalk Server 2006 R2, however, Microsoft has made EDI technologies a first-class citizen in the core product. While Microsoft's partners still have a role to play, such as offering solutions for specific vertical markets, the product itself now provides solid support for B2B integration with EDI.

This support includes several things. First, since different parts of the world use different standards, BizTalk Server 2006 R2 allows using both the X12 standards popular in the United States and the EDIFACT standards that are widely used in Europe and elsewhere. The product also supports the more recent AS/2 standard for exchanging EDI information over the Internet. Along with these, it includes several thousand EDI schemas that support a variety of trading partner formats and requirements, including HIPAA and many others. Organizations can customize these if necessary using Visual Studio. And because the EDI standards define message formats rather than how those messages should be transported, BizTalk Server 2006 R2 implements EDI in pipeline components. This allows any BizTalk adapter to send EDI messages, letting organizations choose the communication approach that's best for them.

Another challenge in B2B integration is managing interactions with trading partners. To make this simpler for EDI connections, BizTalk Server 2006 R2 includes a *Partner Agreement Manager (PAM)*. The PAM allows a BizTalk administrator to configure a variety of settings for each trading partner. For example, different organizations wrap EDI transactions in different ways, use different options for acknowledging these transactions, and batch them together differently. The PAM allows these and other options to be set in whatever way a particular partner requires. BizTalk Server 2006 R2 also includes specialized BAM support that makes it easier for EDI applications to generate BAM data. And to help administrators and other keep track of what's going on, the product provides a set of EDI-oriented reports that can be accessed via the Group Hub page.

Pundits have long predicted the demise of EDI. Yet while no one can argue that it represents the technical state of the art, EDI clearly provides significant business value, and its popularity is still growing. The EDI capabilities in BizTalk Server 2006 R2 are a clear reflection of this reality.

Working with RFID

Radio frequency identification (RFID) offers plenty of potential. An RFID tag is a small device that can be attached to pretty much anything: pallets in a warehouse, individual items in a store, livestock on a farm, passports, and more. This tag contains information that can be read by a nearby RFID reader, then used by applications in any number of ways.

These applications depend on a platform that's capable of accessing and working with RFID data. Microsoft provides this through an RFID server that's part of BizTalk Server 2006 R2. As Figure 14 suggests, this server can be installed and used independently from the other parts of the product. As described later, however, applications built on the BizTalk RFID server will frequently choose to use other parts of BizTalk Server 2006 R2 as well.
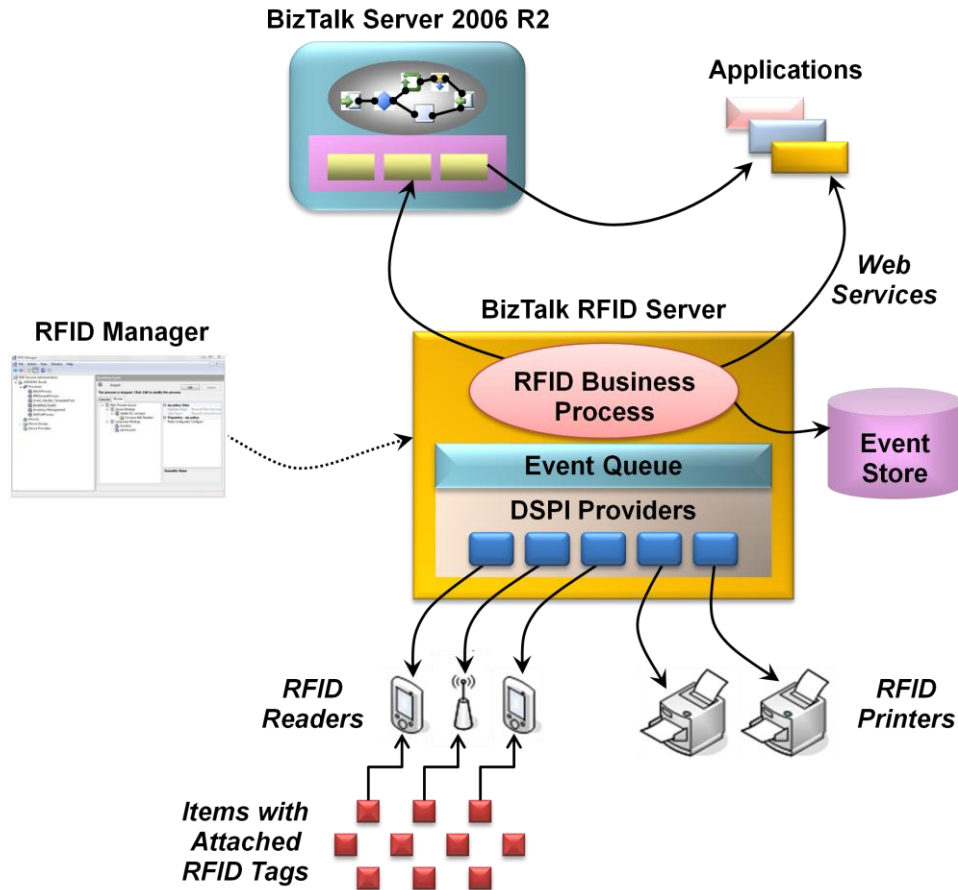
Figure 12: The BizTalk RFID server provides a common platform for RFID applications to interact with diverse RFID devices such as readers and printers.

Items with attached RFID tags—pallets, passports, or whatever—are shown as red squares at the bottom of Figure 14. Their unique identifiers can be read by handheld readers or by fixed devices, such as a reader mounted on the door of a loading dock. These tags might be created by RFID printers, which can produce paper labels with embedded RFID tags, or perhaps in another way.

Many vendors offer RFID readers and RFID printers today, and the hardware they use is quite different. The BizTalk RFID server provides a standard *Device Service Provider Interface (DSPI)* that lets applications work with this diversity in a common way. RFID hardware manufacturers can use a DSPI SDK to create *DSPI providers* for their products. Analogous to a device driver in an operating system (although they contain only user-level code), each provider interacts with a particular kind of device, then exposes its services in a common way through DSPI.

Reading an RFID tag is an event in the real world. Moving a pallet through a loading dock door might trigger a read of that pallet's RFID, for example, as might scanning a passport or herding a cow onto a truck. Whatever kind of device they come from, the BizTalk RFID server places these events into an event queue. A developer can then create an RFID business process, structured as one or more event handlers, that handles these events. To make creating these processes easier, BizTalk Server 2006 R2 provides a new project type for Visual Studio that's focused on building RFID business processes.

This process can do whatever it likes with each event—it's ordinary .NET code written in a language such as C# or Visual Basic. The process might choose to ignore some events, for example, log others in a local event store, and actively respond to only a few. When an RFID business process does

respond to an event, that response will often take the form of a rule: If this event happens, then perform that action. To make creating this kind of logic easier, developers can use the BizTalk Server BRE. The BizTalk RFID server provides custom vocabularies to help developers create rules for working with RFID.

In most cases, an RFID business process will need to communicate with other applications. (In fact, this need for integration is the primary reason that Microsoft's RFID support is licensed as part of BizTalk Server.) An inventory application might need to be informed of stock changes, for example, or a custom .NET application responsible for managing livestock might wish to keep track of each pig. As Figure 14 shows, the RFID business process can communicate directly with other applications using Web services, and it can also communicate with BizTalk Server 2006 R2. A BizTalk application might use a WCF-based adapter to do this, since communication in the RFID server is based on WCF, or it might use some other approach. In any case, the goal is to provide an effective way to turn low-level events—reading RFID tags—into useful business knowledge.

Finally, as the figure shows, the BizTalk RFID server has its own RFID Manager. Using this tool, an administrator can determine which RFID devices are running, examine and modify parameters on those devices (such as which antennas are active on an RFID reader or what air protocol is used to read RFID tags), and more. Microsoft also provides a management pack for Microsoft Operations Manager 2005 and its successor System Center Operations Manager 2007, allowing the BizTalk RFID server to be monitored and managed remotely.

Enterprise Single Sign-On

A business process that relies on several different applications is likely to face the challenge of dealing with several different security domains. Accessing an application on a Windows system may require one set of security credentials, while accessing an application on an IBM mainframe may require different credentials, such as a RACF username and password. Dealing with this profusion of credentials is hard for users, and it can be even harder for automated processes. To address this problem, BizTalk Server 2006 R2 includes Enterprise Single Sign-On.

Don't be confused—this isn't a mechanism that lets people have one login for all applications. Instead, Enterprise Single Sign-On provides a way to map a Windows user ID to non-Windows user credentials. It won't solve all of an organization's enterprise sign-on problems, but this service can make things easier for business processes that use applications on diverse systems.

To use Enterprise Single Sign-On, an administrator defines *affiliate applications*, each of which represents a non-Windows system or application. An affiliate application might be a CICS application running on an IBM mainframe, an SAP R/3 system running on Unix, or any other kind of software. Each of these applications has its own mechanism for authentication, and so each requires its own unique credentials.

Enterprise Single Sign-On stores an encrypted mapping between a user's Windows ID and his credentials for one or more affiliate applications in a *credential* database. When this user needs to access an affiliate application, his credentials for that application can be looked up in the Credential database by a *Single Sign-On (SSO) Server*. Figure 15 shows how this works.
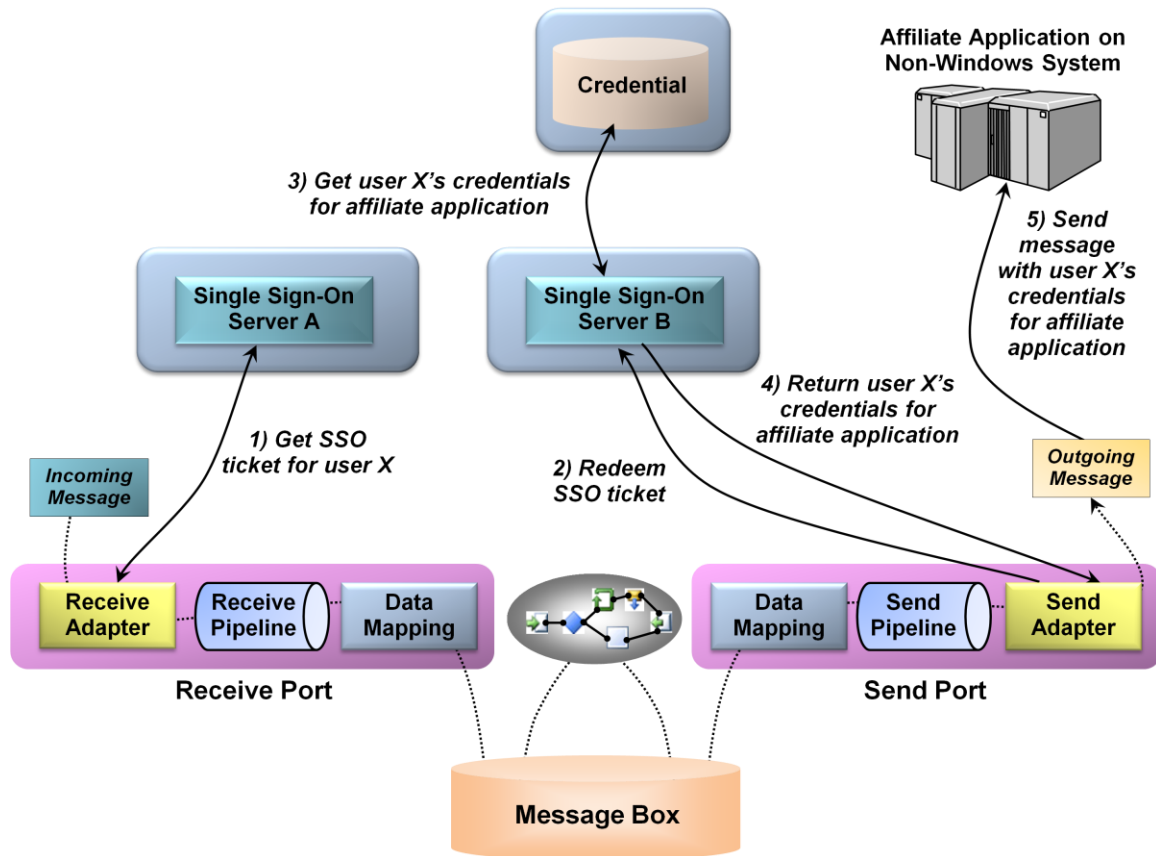
Figure 13: Enterprise Single Sign-On allows mapping between a user's Windows credentials and those required for other systems.

In this example, a message sent to a BizTalk application is processed by an orchestration, then sent to an affiliate application running on an IBM mainframe. The job of Enterprise Single Sign-On is to make sure that the correct credentials (e.g., the right username and password) are sent with the message when it is passed to the affiliate application.

As the diagram shows, when a receive adapter gets a message, the adapter can request an *SSO ticket* from SSO server A (step 1). This encrypted ticket contains the Windows identity of the user that made the request and a timeout period. (Don't confuse this with a Kerberos ticket—it's not the same thing.) Once it's acquired, the SSO ticket is added as a property to the incoming message. The message then takes its normal path through BizTalk Server 2006 R2, which in this example means being handled by an orchestration. When this orchestration generates an outgoing message, that message also contains the SSO ticket acquired earlier.

This new message is destined for the application running on an IBM mainframe, and so it must contain the appropriate credentials for this user to access that application. To get these credentials, the send adapter contacts SSO server B (step 2), supplying the message (which contains the SSO ticket) it just received and the name of the affiliate application it wishes to retrieve the credentials for. This operation, called *redemption*, causes SSO server B to verify the SSO ticket, and then look up this user's credentials for that application (step 3). SSO Server B returns those credentials to the send adapter (step 4), which uses them to send an appropriately-authenticated message to the affiliate application (step 5).

Enterprise Single Sign-On also includes administration tools to perform various operations. All operations performed on the credential database are audited, for example, so tools are provided that allow an administrator to monitor these operations and set various audit levels. Other tools allow an administrator to disable a particular affiliate application, turn on and off an individual mapping for a user, and perform other functions. There's also a client utility that lets end users configure their own credentials and mappings. And like other parts of BizTalk Server 2006 R2, Enterprise Single Sign-On exposes its services through a programmable API. Creators of third-party BizTalk adapters can use this API to access the single sign-on services, while administrators can use it to create scripts for automating common tasks.

The example described above shows a typical use of Enterprise Single Sign-On, but it's not the only option. A smaller BizTalk Server 2006 R2 installation may have only a single SSO server, for example, and it's even possible to use Enterprise Single Sign-On independently from BizTalk Server. Because BizTalk applications typically need to interact with other applications on different systems, including this component as part of the product makes good sense.

Conclusion

The goal of BizTalk Server 2006 R2 is to help organizations create automated business processes that span diverse applications and platforms. Along with its core messaging and orchestration capabilities, the product includes a BRE for working with complex business rules and BAM to let information workers track running processes. Additional components, such as EDI support, the RFID server, and Enterprise Single Sign-On, address other related challenges.

From its initial roots in EAI and B2B integration, BizTalk Server has grown into a foundation for BPM. As the change to a service-oriented world rolls on, BizTalk Server 2006 R2 will continue to play an important part in automating business processes in a diverse world.

About the Author

David Chappell is Principal of Chappell & Associates (www.davidchappell.com) in San Francisco, California. Through his speaking, writing, and consulting, he helps information technology professionals understand, use, and make better decisions about enterprise software.